ADVANCED SQL PROJECT

FOOTBALL ANALYTICS

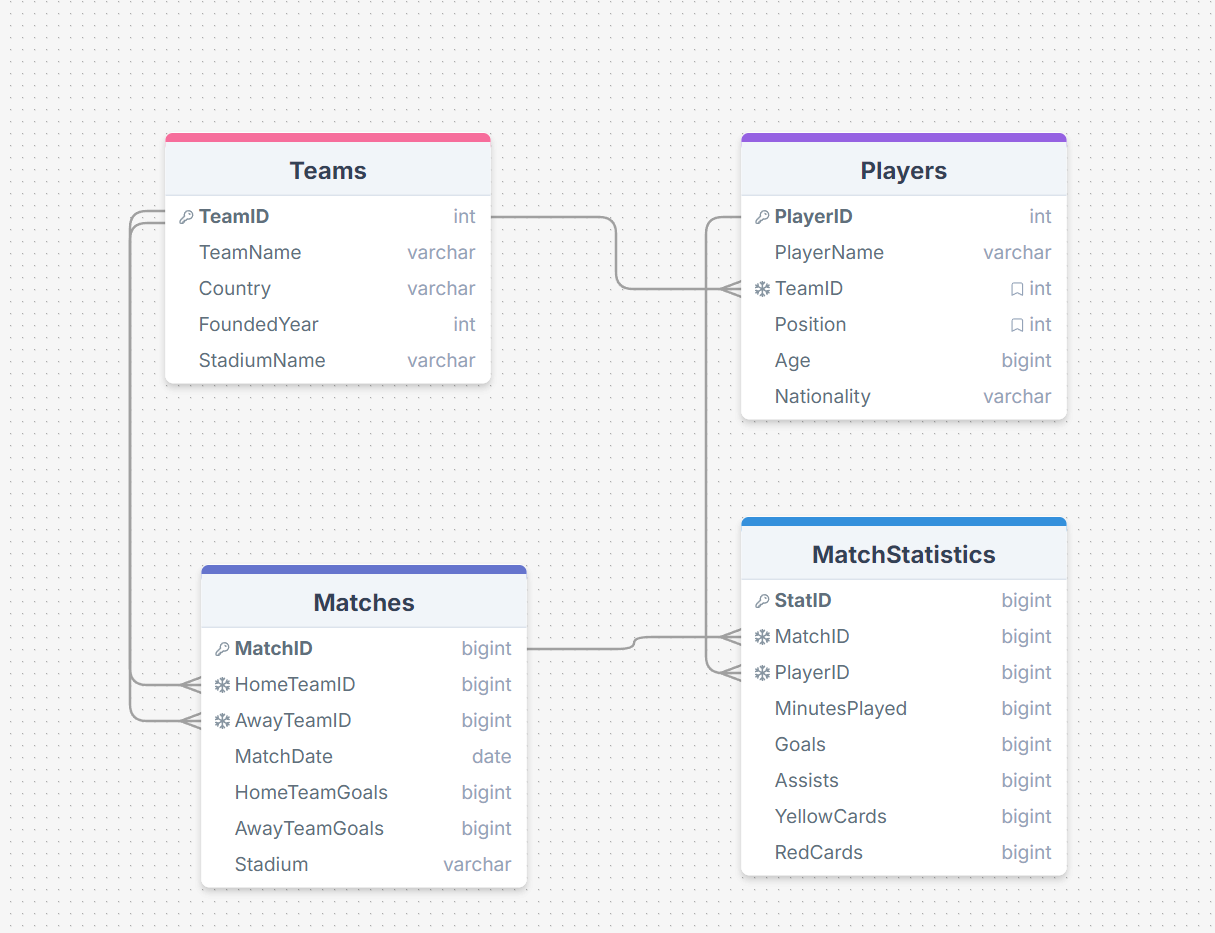
TEACHER: [CORBEA ALEXANDRA-MARIA-IOANA](https://online.ase.ro/user/view.php?id=4209&course=1)

STUDENT: COJOCARU FLORIN-VALENTIN

STUDY GROUP NUMBER: 1139

* **Conceptual schema of the database**

As a first step in building the project, I have started by designing the conceptual schema, or entity-relationship diagram, for a football database. This database is designed to manage and organize data related to football teams, players, matches, and match statistics. The schema consists of four main entities: Teams, Players, Matches, and MatchStatistics, each capturing essential details relevant to football operations.



In brief, the relationships between the tables in the football database schema are as follows:

* **Teams and Players**: There is a **one-to-many** relationship between the Teams and Players tables, meaning each team can have multiple associated players.
* **Teams and Matches**: There is a **one-to-many** relationship between Teams and Matches through the HomeTeamID and AwayTeamID fields, indicating that each team can participate in multiple matches, either as the home team or the away team.
* **Matches and MatchStatistics**: There is a **one-to-many** relationship between the Matches and MatchStatistics tables, allowing each match to have multiple statistics recorded for different players.
* **Players and MatchStatistics**: There is also a **one-to-many** relationship between Players and MatchStatistics, indicating that a player can have multiple entries in match statistics for different matches.
* **DATABASE AND TABLES BUILDING IN SQL DEVELOPER**

CREATE TABLE Teams (

TeamID NUMBER PRIMARY KEY,

TeamName VARCHAR2(100) NOT NULL,

Country VARCHAR2(50),

FoundedYear NUMBER,

StadiumName VARCHAR2(100)

);

CREATE SEQUENCE TeamID\_seq START WITH 1 INCREMENT BY 1;

CREATE TABLE Players (

PlayerID NUMBER PRIMARY KEY,

PlayerName VARCHAR2(100) NOT NULL,

TeamID NUMBER,

Position VARCHAR2(20),

Age NUMBER,

Nationality VARCHAR2(50),

FOREIGN KEY (TeamID) REFERENCES Teams(TeamID)

);

CREATE SEQUENCE PlayerID\_seq START WITH 1 INCREMENT BY 1;

CREATE TABLE Matches (

MatchID NUMBER PRIMARY KEY,

HomeTeamID NUMBER,

AwayTeamID NUMBER,

MatchDate DATE,

HomeTeamGoals NUMBER,

AwayTeamGoals NUMBER,

Stadium VARCHAR2(100),

FOREIGN KEY (HomeTeamID) REFERENCES Teams(TeamID),

FOREIGN KEY (AwayTeamID) REFERENCES Teams(TeamID)

);

CREATE SEQUENCE MatchID\_seq START WITH 1 INCREMENT BY 1;

CREATE TABLE MatchStatistics (

StatID NUMBER PRIMARY KEY,

MatchID NUMBER,

PlayerID NUMBER,

MinutesPlayed NUMBER,

Goals NUMBER,

Assists NUMBER,

YellowCards NUMBER,

RedCards NUMBER,

FOREIGN KEY (MatchID) REFERENCES Matches(MatchID),

FOREIGN KEY (PlayerID) REFERENCES Players(PlayerID)

);

CREATE SEQUENCE StatID\_seq START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER trg\_Teams

BEFORE INSERT ON Teams

FOR EACH ROW

BEGIN

SELECT TeamID\_seq.NEXTVAL INTO :NEW.TeamID FROM dual;

END;

/

CREATE OR REPLACE TRIGGER trg\_Players

BEFORE INSERT ON Players

FOR EACH ROW

BEGIN

SELECT PlayerID\_seq.NEXTVAL INTO :NEW.PlayerID FROM dual;

END;

/

CREATE OR REPLACE TRIGGER trg\_Matches

BEFORE INSERT ON Matches

FOR EACH ROW

BEGIN

SELECT MatchID\_seq.NEXTVAL INTO :NEW.MatchID FROM dual;

END;

/

CREATE OR REPLACE TRIGGER trg\_MatchStatistics

BEFORE INSERT ON MatchStatistics

FOR EACH ROW

BEGIN

SELECT StatID\_seq.NEXTVAL INTO :NEW.StatID FROM dual;

END; /

* **Hierarchical Queries**

Display the team hierarchy, showing parent-child relationships among teams

SELECT LPAD(' ', 2 \* (LEVEL - 1)) || TeamName AS Hierarchy, TeamID

FROM Teams

CONNECT BY NOCYCLE PRIOR TeamID = TeamID

START WITH TeamID IS NOT NULL;

****

List players and their direct relationships with matches in a hierarchical format.

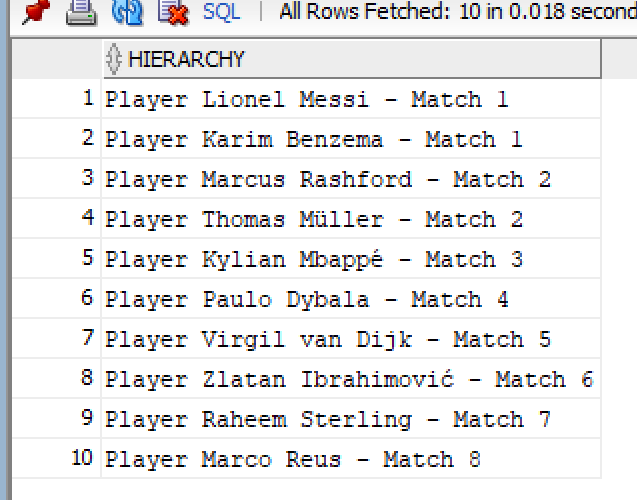
SELECT LPAD(' ', 2 \* (LEVEL - 1)) || 'Player ' || p.PlayerName || ' - Match ' || ms.MatchID AS Hierarchy

FROM Players p

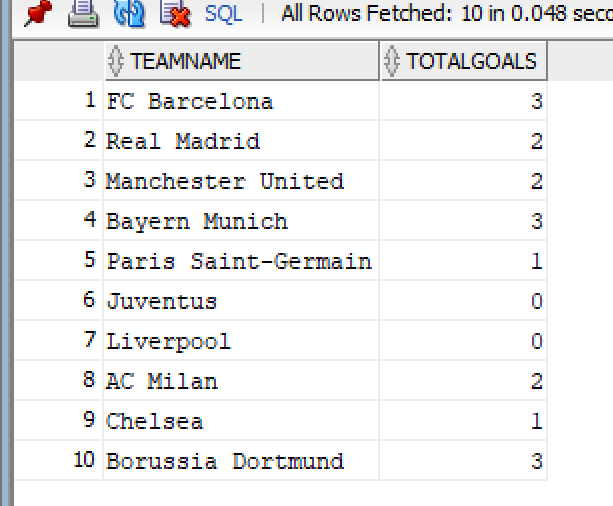
JOIN MatchStatistics ms ON p.PlayerID = ms.PlayerID

CONNECT BY PRIOR ms.MatchID = ms.MatchID

START WITH p.PlayerID IS NOT NULL;



* **Complex Queries with Joins, Subqueries, and Aggregations**

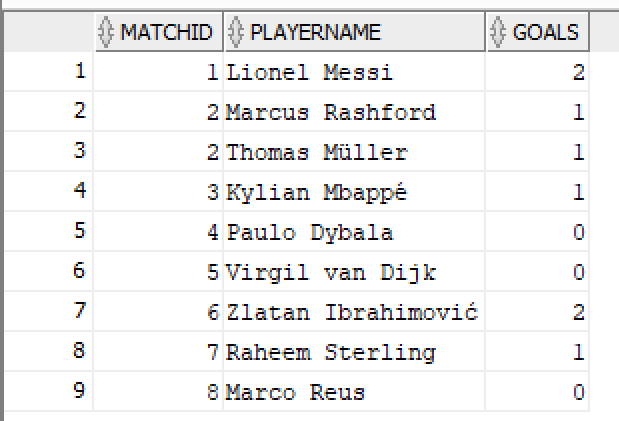
1. Find total goals scored by each team in home matches.

SELECT t.TeamName, SUM(m.HomeTeamGoals) AS TotalGoals

FROM Matches m

JOIN Teams t ON m.HomeTeamID = t.TeamID

GROUP BY t.TeamName;

2. Identify the player who scored the highest number of goals in each match.

SELECT ms.MatchID, p.PlayerName, ms.Goals

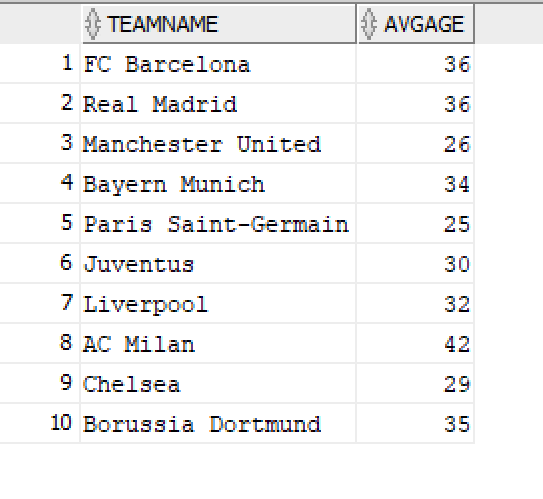
FROM MatchStatistics ms

JOIN Players p ON ms.PlayerID = p.PlayerID

WHERE ms.Goals = (SELECT MAX(ms2.Goals)

FROM MatchStatistics ms2

WHERE ms2.MatchID = ms.MatchID);



3. List teams and their average player age.

SELECT t.TeamName, AVG(p.Age) AS AvgAge

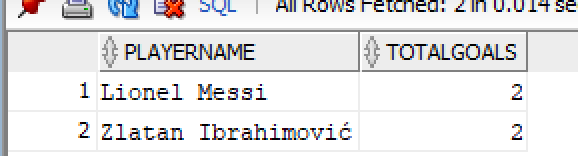
FROM Players p

JOIN Teams t ON p.TeamID = t.TeamID

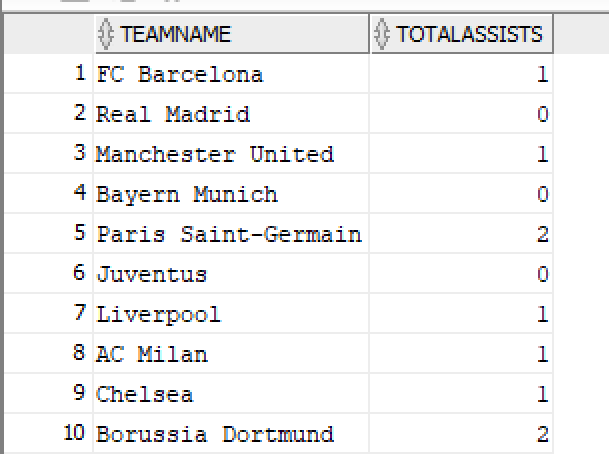
GROUP BY t.TeamName;

4. Display players with more than 1 total goal scored.

SELECT p.PlayerName, SUM(ms.Goals) AS TotalGoals FROM Players p

JOIN MatchStatistics ms ON p.PlayerID = ms.PlayerID GROUP BY p.PlayerName

HAVING SUM(ms.Goals) > 1 ;

5. Find the total number of assists made by players for each team.

SELECT t.TeamName, SUM(ms.Assists) AS TotalAssists

FROM MatchStatistics ms

JOIN Players p ON ms.PlayerID = p.PlayerID

JOIN Teams t ON p.TeamID = t.TeamID

GROUP BY t.TeamName;

6. List matches where the home team scored more than two goals, using subqueries.

SELECT m.MatchID, t.TeamName, m.HomeTeamGoals

FROM Matches m

JOIN Teams t ON m.HomeTeamID = t.TeamID

WHERE m.HomeTeamGoals > 2 AND m.MatchID IN (SELECT MatchID FROM Matches WHERE HomeTeamGoals > 2);

7. Get the number of matches played by each team.

SELECT t.TeamName, COUNT(m.MatchID) AS MatchesPlayed

FROM Matches m

JOIN Teams t ON m.HomeTeamID = t.TeamID OR m.AwayTeamID = t.TeamID

GROUP BY t.TeamName;

* **Analytical Function Queries**



1.Rank players by total goals scored.

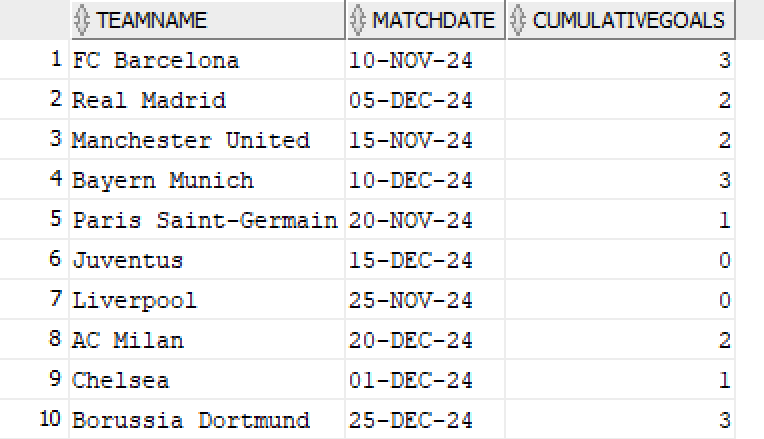
SELECT p.PlayerName, SUM(ms.Goals) AS TotalGoals,

RANK() OVER (ORDER BY SUM(ms.Goals) DESC) AS GoalRank

FROM Players p

JOIN MatchStatistics ms ON p.PlayerID = ms.PlayerID

GROUP BY p.PlayerName;



2.Show the cumulative goals scored by teams in matches.

SELECT t.TeamName, m.MatchDate,

SUM(m.HomeTeamGoals) OVER (PARTITION BY t.TeamID ORDER BY m.MatchDate) AS CumulativeGoals

FROM Matches m

JOIN Teams t ON m.HomeTeamID = t.TeamID;

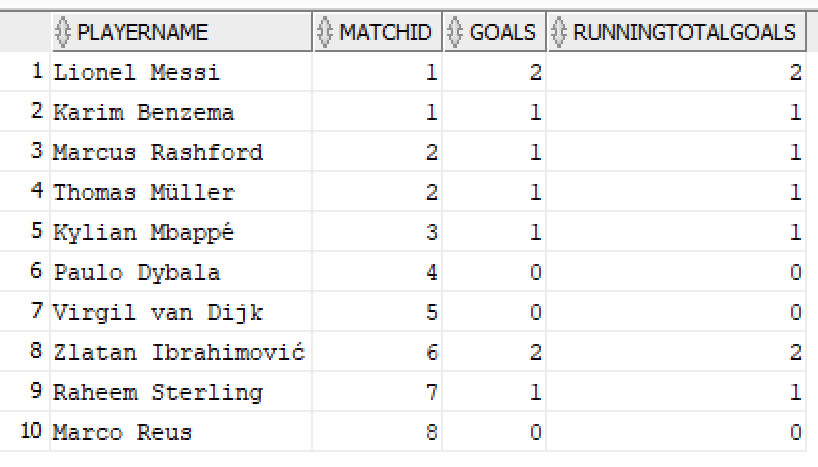
3. Calculate the average goals per game for each player using analytical functions.

SELECT p.PlayerName,

AVG(ms.Goals) OVER (PARTITION BY p.PlayerID) AS AvgGoalsPerGame

FROM Players p

JOIN MatchStatistics ms ON p.PlayerID = ms.PlayerID;

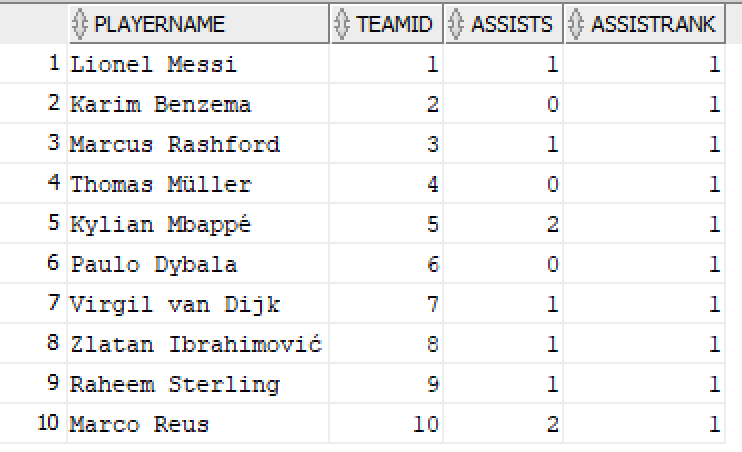
4. Display the change in match attendance over time for each team.

SELECT p.PlayerName, ms.MatchID, ms.Goals,

SUM(ms.Goals) OVER (PARTITION BY p.PlayerID ORDER BY ms.MatchID) AS RunningTotalGoals

FROM Players p

JOIN MatchStatistics ms ON p.PlayerID = ms.PlayerID;

5. Rank players by the number of assists they provided within their team.

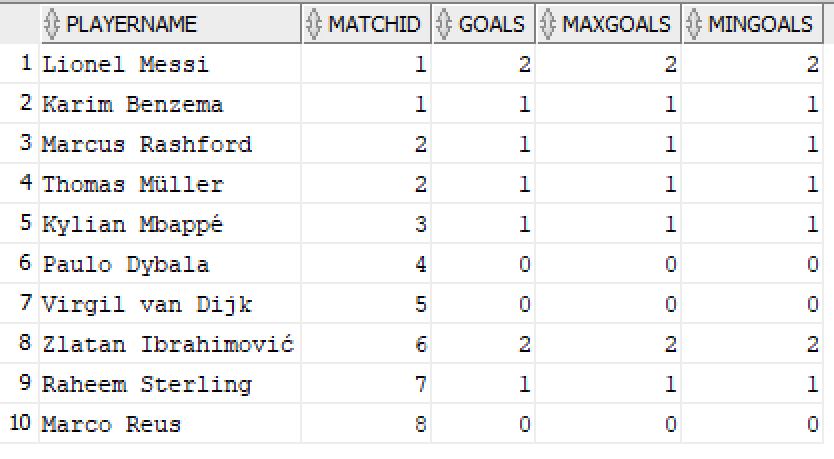
SELECT p.PlayerName, p.TeamID, ms.Assists,

RANK() OVER (PARTITION BY p.TeamID ORDER BY ms.Assists DESC) AS AssistRank

FROM Players p

JOIN MatchStatistics ms ON p.PlayerID = ms.PlayerID;

6. Find the highest and lowest number of goals scored by each player in matches.

SELECT p.PlayerName, ms.MatchID, ms.Goals,

MAX(ms.Goals) OVER (PARTITION BY p.PlayerID) AS MaxGoals,

MIN(ms.Goals) OVER (PARTITION BY p.PlayerID) AS MinGoals

FROM Players p

JOIN MatchStatistics ms ON p.PlayerID = ms.PlayerID;